UNITED STATES PATENT APPLICATION

FOR

SYSTEM AND METHOD FOR DYNAMICALLY GENERATING A CONFIGURATION DATASHEET

Inventors:

DOUG ANDERSON
KEN OGAMI

Prepared by:
WAGNER, MURABITO & HAO LLP
Two North Market Street
Third Floor
San Jose, California 95113

CYPRC01174M/ACM/MDM

SYSTEM AND METHOD FOR DYNAMICALLY GENERATING A CONFIGURATION DATASHEET

FIELD OF THE INVENTION

Embodiments of the present invention relate to the field of programming environments for configurable integrated circuits. More particularly, embodiments of the present invention relate to a method for generating configuration datasheets for the integrated circuit being configured.

BACKGROUND ART

Traditionally, integrated circuit design and fabrication have involved the efforts of a team of people in producing a finished part. Design, prototyping, testing and final productization have usually required more than one iteration, and the design task has been largely separated functionally and temporally, from the process of final documentation. Device data would typically be collected from various sources and transferred to an individual such as a technical writer, who would prepare a device datasheet. Among the sources of the data would be a designer, manually compiling a list of data.

The advent of programmable integrated circuits and computer-aided design (CAD) tools has reduced the number of individuals required to carry a concept from design to productization. The development of the integrated design environment (IDE) has made it possible for individual engineers to configure a programmable integrated circuit.

The IDE has made it feasible for a single designer to configure sophisticated integrated circuits with dozens of pins and many possible inputs and outputs. An example of such integrated circuits is the programmable microcontroller. A programmable microcontroller may include a microprocessor, memory, and digital and other programmable hardware resources. The number of pins and possible signals significantly increase the amount of information required to document the final product. Manual compilation of data by a designer is tedious and prone to error.

Although many tasks have been efficiently combined into current integrated design environments, the task of final documentation has not. Final layout and generation of a product datasheet still requires the transfer of data to an external capability, and manual compilation and editing, e.g., using a word processor and/or other document editor. This required data transfer increases the chance for error and the time required to produce a datasheet.

SUMMARY OF INVENTION

Accordingly, embodiments of the present invention provide a method and system for generating a datasheet from within a computer aided integrated circuit design environment. Embodiments of the present invention further include a database structure optimized for the production of graphics and textual output such as a datasheet.

A system and method for dynamically generating a project configuration datasheet are disclosed. In one embodiment, Module and device descriptions are stored in extensible markup language (XML) format. The IDE includes an XSL (extensible stylesheet language) stylesheet in one example. The module and device descriptions are combined with parameterization information as prescribed by the XSL stylesheet to automatically produce a project configuration report by computer processes. The project configuration report may be formatted in hypertext markup language (HTML) and may be rendered as a visual datasheet by a browser.

In an embodiment of the present invention, an integrated design environment for a programmable system-on-a-chip stores the descriptions of the modules and devices available for configuration by the designer in a database that is formatted with extensible markup language (XML). The parameterization information is combined with the module and device descriptions to automatically produce a project configuration report. This combination is performed using an extensible stylesheet language (XSL) stylesheet. XSL may be used to transform the XML data into an HTML (hypertext markup language) file that is then rendered by a browser.

Another embodiment of the present invention includes a computer system with a printer or a visual display or other document rendering mechanism that includes an integrated design environment (IDE) software application. Information concerning the modules and devices available to the designer is stored in an XML formatted database. In the course of developing a design, the design parameters are generated and linked to the devices and modules. The computer system also includes a stored XSL stylesheet that governs the transformation and formatting of the XML data. The computer system also includes a browser for rendering HTML as visual output, either to a display device or to a printer.

These and other objects and advantages of the present invention will no doubt become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1A is a diagram of a computer system portion of an environment in which embodiments of the present claimed invention may be practiced.

Figure 1B shows a device editor window of an integrated design environment used for user module selection in accordance with an embodiment of the present claimed invention.

Figure 1C shows a device editor window of an integrated design environment used for user module placement in accordance with an embodiment of the present claimed invention.

Figure 1D shows integrated design environment window used for specification of module parameters and global resources in accordance with an embodiment of the present claimed invention.

Figure 1E shows a device editor window of an integrated design environment used for specifying device pinout accordance with an embodiment of the present claimed invention.

Figure 2A shows a flow chart for the generation of project configuration data in accordance with an embodiment of the present claimed invention.

Figure 2B shows a data flow diagram for the generation of a datasheet in accordance with an embodiment of the present claimed invention.

Figures 3A, 3B, 3C, 3D, 3E, 3F and 3G show an example of portions of a database formatted in XML in accordance with an embodiment of the present claimed invention.

Figures 4A, 4B, 4C, 4D, 4E, 4F, 4G, 4H, and 4I illustrate an XSL stylesheet in accordance with an embodiment of the present claimed invention.

Figures 5A, 5B, 5C, 5D and 5E show a visual output datasheet in accordance with an embodiment of the present invention as rendered by a browser.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following detailed description of the present invention, a system and method for dynamically generated configuration datasheet; numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one skilled in the art that the present invention may be practiced without these specific details. In other instances well known methods, procedures, components, and circuits have not been described in detail so as not to unnecessarily obscure aspects of the present invention.

Notation and Nomenclature

Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is here, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the disclosure of the present invention, terms such as "processing" or "computing" or "calculating" or "computing" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system's registers or memories or other such information storage, transmission or display devices.

Refer to Figure 1A which illustrates a computer system 112. In general, computer systems 112 used by the preferred embodiment of the present invention comprise a bus 100 for communicating information, a central processor 101 coupled with the bus for processing information and instructions, a random access memory 102 coupled with the bus 100 for storing information and instructions for the central processor 101, a read only memory 103 coupled with the bus 100 for storing static information and instructions for the processor 101, a data storage device 104 such as a magnetic or optical disk and disk drive coupled with the bus 100 for storing information and instructions, a display device 105 coupled to the bus 100 for displaying information to the computer user, an alphanumeric input device 106 including alphanumeric and function keys coupled to the bus 100 for communicating user input information and command selections to the central processor 101, cursor control device 107 coupled to the bus for communicating user input information and command selections to the central processor 101, and a communications port 108 coupled to the bus 100 communication with the system 112. The port 108 may be coupled to printer.

The display device 105 of Figure 1A utilized with the computer system of the present invention may be a liquid crystal device, cathode ray tube or other display device suitable for creating graphic images and alphanumeric characters recognizable to the user. The cursor control device 107 allows the computer user to dynamically signal the two dimensional movement of a visible symbol (pointer) on a display screen of the display device 105. Many implementations of the cursor control device are known in the art including a trackball, mouse, joystick or special keys on the alphanumeric input device 105 capable of signaling movement of a given direction or manner of displacement. It is to be appreciated that the cursor means 107 also may be directed and/or activated via input from the keyboard using special keys and key sequence commands. Alternatively, the cursor may be directed and/or activated via input from a number of specially adapted cursor directing devices.

The computer system 112 of Figure 1A may be used with an integrated design environment (IDE). An example of an IDE is the Cypress PSoC Designer. PSoC Designer contains three subsystems; Device Editor, Application Editor, and Debugger. The interface is split into several active windows that differ depending on which subsystem you are in.

The PSoC Designer Device Editor of Figure 1B is used for selection, placement and configuration of User Modules. Figure 1B shows a device editor interface 120 in selection mode. The Device Editor is the subsystem in which the bulk of datasheet information is generated. Shown in Figure 1B are a User Module Selection Window 121 User Module Window 122, User Information Windows 123, and a Resource Manager Window 124.

In an embodiment of the invention a project begins with the selection of a target device such as the Cypress Microsystems 8C25122A microcontroller. The 8C25122A is a member of the CY8C25xxx/26xxx family of Programmable System-on-Chip (PSoC) microcontrollers that replaces many MCU-based system components with one single-chip, programmable device. A

single PSoC microcontroller offers a fast core, Flash program memory, and SRAM data memory with configurable analog and digital peripheral blocks in a range of convenient pin-outs and memory sizes. The driving force behind this innovative programmable system-on-a-chip comes from user configurability of analog and digital arrays, the PSoC blocks. PSoC blocks, are analog and digital peripheral blocks that are customized by the placement and configuration of User Modules.

The Device Editor interface 120 is used to configure the target device. User Modules (represented by icons) may be selected from the User Module Window 122, which causes an icon to appear in the User Module Selection window 121. In this example a pulse width modulator PWM16_1 125 is shown as selected. Information regarding the PWM16_1 125 is shown in the User Information Windows 123. The Resource Manager Window shows the target device resources that are used by the PWM16_1.

Figure 1C shows the device editor interface 120 in placement mode, with a Placement Window 130, a Global Resources Window 131, and a User Module Parameters Window 132. User Modules shown in the User Module Selection window 121 are placed in the Placement Window 130. The placement of the PWM16_1 125 is indicated by the two digital blocks 133 and 134 shown in the Placement Window 130. Parameters and resources shown in the Global Resources Window 131 and User Module Parameters Window 132 are available for configuration by the designer

Figure 1D shows the Device Editor interface 120 with expanded Global Resources Window 131 and User Module Parameter Window in which resources and parameters have been set. For example, in the Global Resources Window 131, the CPU_Clock has been set with a frequency of 12 MHz and the Sleep Timer has been set with a frequency of 512 Hz. In the User Module Parameters Window the period has been set at 134 and the Pulse Width has been set at 92. The values entered by the designer, in addition to default values, are stored in an XML formatted database.

Figure 1E shows a Device Editor interface of an integrated design environment used for deploying module connections. Interconnections can be specified on the device in the placement mode of Device Editor. User Module interconnections consist of connections to surrounding PSoC blocks, output bus, input bus, internal system clocks and references, external pins, and analog output buffers. Multiplexors may also be configured to route signals throughout the PSoC block architecture.

Referring again to Figure 1E, a Pinout Window 140 and a Pinout Parameters Window 141 are shown. The Pinout window includes a diagram of the pin layout of the target device (e.g., the Cypress Microsystems 8C25122A). The Pinout Window accepts input specifying the connections for the PSoC blocks to the pins. In this example, there is only one User Module present, and thus there are no interconnections specified between multiple User Modules. Typically, there may be multiple User Modules with designer specified interconnects which would be stored in the project database. Some interconnects are designer specified, whereas others are generated automatically (e.g., the interconnection of PSoC blocks of a selected User Module.)

Figure 2A shows a flow chart 200 for the generation of data for a project configuration datasheet in accordance with an embodiment of the present claimed invention. This generation of data may be performed with system similar to that referred to in Figures 1A through 1E.

In step 205, a target device is selected. An integrated design environment may be dedicated to a single device, in which case the selection is automatic. Otherwise, the selection is made by a user.

In step 210, User Modules representing circuit designs are selected. The circuit designs represented by the User Modules correspond to combinations of analog and digital blocks available on the target device. The blocks are part of a plurality of programmable hardware resources.

In step 215, the User Modules are placed. The placement of the User Modules is done in accordance with the allowable combinations available for the blocks. Since redundant blocks may be available, more than one possible combination may be allowed.

In step 220, the User Module and Global Resource parameters are selected. Parameters are typically selected by the designer from allowable values and may or may not have defaults established by the design environment.

In step 225, connections between User Modules is specified. If more than one User Module is present, the designer may specify interconnections between the modules.

In step 230, the pinout connections are specified. In this step, the connections between the circuit blocks on the target device and the physical pins of the package are established.

Figure 2B shows a flow chart 240 for the automatic generation of a project configuration datasheet in accordance with an embodiment of the present claimed invention.

Project data 245 is obtained from a database that is formatted in an extensible markup language (XML). In conventional design environments, information is typically stored in structures that emphasize accessibility and compact storage. In the present invention, XML is used to facilitate the visual rendering of information. The XML formatted database is adapted to accept the parameter values as determined by the designer, and thus a complete descriptive file of parameter names and values is available in XML format.

Generally, Extensible Markup Language (XML) can be used as a human-readable, machine-understandable, general syntax for describing hierarchical data. XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form the character data in the document, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure. A software module called an XML processor is used to read XML documents and provide access to their content and structure.

The XML formatted database can accommodate in its tree structure any project information generated by the activities of the designer.

In the data flow diagram 240 of Figure 2B, An XSL stylesheet 250 is selected as input to an XSL/XML processor 255, along with project configuration data in XML format 245. The project configuration data in XML format 245 may be data generated by the process shown in Figure 2A. Generally, a stylesheet describes rules for presenting a class of XML source documents and is expressed in extensible stylesheet language (XSL). XSL is composed of a language for transforming XML documents and an XML vocabulary for specifying formatting semantics. An XSL stylesheet specifies the presentation of a class of XML documents (e.g., user module data) by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary.

The XSL stylesheet 250 and XML formatted data 245 are processed by an XSL/XML processor 255 to produce a project datasheet file 260. The XSL/XML processor applies the instructions of the stylesheet 250 to the XML formatted data

245. The combination of a XSL stylesheet and the XML data is called a project datasheet file (or configuration report). The project datasheet file 260 contains all of the information necessary for visual presentation of the XML data as well as elements provided by the XSL stylesheet (e.g., graphics images.) There are two aspects of the presentation process. first, a result tree is constructed from the XML source tree, that is, a tree transformation is carried out. Tree transformation allows the structure of the result tree to be significantly different from the source tree.

A browser 265 may be used to render the project datasheet file 260 as a visible datasheet 270. An example of visual rendering is the use of a browser to display the project datasheet 260 on a display device or have it printed on a printer. XSL is capable of transforming XML to HTML. A browser such as NETSCAPE NAVIGATOR or MICROSOFT INTERNET EXPLORER may be used to render a visual datasheet 270 from the datasheet file 260. It should be noted that the datasheet file (or configuration report) 260 is an electronic embodiment of the present invention, and that the rendered output 270 is a visual embodiment.

Figures 3A through 3G show an example of a portion of a database formatted in XML in accordance with an embodiment of the present claimed invention. The XML formatted data can be associated with a device such as the Cypress Microsystems 8C25122A programmable system-on-a-chip (PSoC)and a computer aided design system, e.g., Cypress Microsystems PSoC Designer. The part includes a microprocessor, memory, digital blocks, analog blocks, and 8 pins (6 I/O pins.) Examples of the data that may be included in the database are pin-out information, schematics, connectivity, parameters, block information, and signal information. Hardware suitable for use with the present invention is described in a U.S. Patent Application titled "Microcontroller Programmable System on A Chip" by W. Snyder, filed on October 22, 2001; the whole of which is incorporated herein by reference. Additionally, an environment in which the present invention may be practiced is described in U.S. Patent Applications No. 09/972,003 (filed October 5, 2001), No. 09/972,133 (filed October 5, 2001), and No. 09/972,319; which are incorporated herein by reference.

The hierarchical nature of the PSoC is well suited to representation in an XML formatted database, and the XML formatted data is readily available as a direct result of the design activity. The data that serves as the basis for the automatic generation of a datasheet does not require a exceptional overhead or a specialized processor.

In the example of Figures 3A through 3G, the description and parameters associated with a design project having a single user module, a PWM16, is shown.

Figure 3A shows the User Module List 31 in which the registers associated with the PWM16 are specified. Figures 3B through 3E show parameter and resource values (e.g., the selectable values for the Global Resources shown in the Global Resources Window 131 of Figure 1D corresponding to Figure 3B.) Some Global Resource values, such as the Reserved Resource list of Figure 3B through 3F may not be user selectable. Figure 3G shows the pinout data 32 associated with the user selections made through the Design Editor interface of Figure 1E.

Figures 4A through 4I illustrate an XSL stylesheet in accordance with an embodiment of the present claimed invention. Unlike HTML, element names in XML have no intrinsic presentation semantics. Without a stylesheet, a processor could not intelligibly present the XML data of Figures 3A through 3G. However, the use of XSL allows a single stylesheet to handle many different permutations of the XML data generated by a designer. Thus, it is possible to use a single stylesheet to handle all of the possible project datasheets, and just a few stylesheets are capable of providing considerable flexibility in the final output in terms of style and composition (.e.g. substitution of color images for black and white in embedded images).

Figure 4A shows the XSL used for formatting the appearance and alignment (font, size, weight, etc.) of the data associated with the Base Device (CY8C25122-24PI) specified in the XML data of Figure 3A. Similarly, Figures 4B through 4I contain the instructions for transforming the remainder of the XML formatted data. In addition to the transformation of the data shown in Figures 3A through 3G, the stylesheet may add other elements to the output datasheet file. For example, the icon.gif 41 and part.jpg 42 graphics files specified in figure 4B. A single stylesheet may be used to produce a wide array of output datasheet files by operating on different sets of XML formatted data.

Figures 5A through 5E shows a visual output datasheet as rendered by a browser. From within an IDE, the visual output may be obtained by simply clicking on an icon (invoking a browser for the datasheet of the current project) or clicking on a datasheet file in a directory (associating the datasheet file type with the browser application.) Either of these methods is referred to as a "single action datasheet display"

Since the project configuration data is stored in an XML formatted database, the overhead involved in creating a datasheet file using an XSL stylesheet is minimal. A datasheet file may be updated continuously and automatically with each configuration action performed by a designer.

Figure 5A shows a portion of the data provided by interaction with the Device Editor interface shown in figures 1B through 1E. The Signal Pin Table 503 of Figure 5A is produced from input to the Pinout Window 140 and the Pinout Parameters Window 141 of Figure 1E. Similarly, the Selected Global Parameters table 505 of Figure 5A originate from input provided using Global Resource Window 131 shown in Figure 1D.

Other user selected input appears in the Parameters table 507 of Figure 5B and the Pin table 509 of Figure 5D.

In addition to the user selected input, the visual output datasheet also includes graphics elements such as the PWM schematic 506 shown in Figure 5B and the 25122PDIP outline 504 in the Signal Pin Table 503 of Figure 5A. Other graphics elements are the logo 500 and icon 502 shown in Figure 5A. Graphics elements may be retrieved on the basis of information referenced in the database, but are not necessarily a part of the database itself. A graphics image such as the "Cypress Microsystems" logo 500 at the top of Figures 5A, 5B and 5D may be inserted by the stylesheet without reference to the XML formatted data.

The visual output datasheet also displays information not explicitly input using the Device Editor interface, such as the Blocks Table 508 of Figure 5C and the Global Register Values table 513 in Figure 5E. Other data tables that may be included are the Analog Clocks table 510 and Analog Input MUX table 511 of Figure 5D, and the Analog Buffer Output table 512 of Figure 5E. Also shown in Figure 5A is a file information block 501, containing information regarding the generation of the datasheet file.

The XML formatted database and application of XSL stylesheets of the present invention are a further step in the integration of the overall product design effort. The task of producing visually descriptive information about a project or a product is automated by the application of XML to the fundamental data structures used in the design environment, and by the use of XSL to provide for the transformation of fundamental design data from the electronic state to a visually coherent presentation.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.